1階述語言語の逐次漸近標準形について(数式処理と数学研究への応用)

AUTHOR(S):

元吉, 文男; 佐藤, 泰介

CITATION:

元吉, 文男 ...[et al]. 1階述語言語の逐次漸近標準形について(数式処理と数学研究への応用). 数理解析研究所講究録 1991, 753: 135-143

ISSUE DATE:

1991-05

URL:

http://hdl.handle.net/2433/82079

数理解析研究所講究録
第753巻 1991年 135-143

1階述語言語の逐次漸近標準形について

On Incremental Asymptotic Normal Form for First Order Language

元吉文男 ・ 佐藤泰介
Fumio Motoyoshi ・ Taisuke Sato

電子技術総合研究所
ElectroTechnical Laboratory

1. はじめに

論理型言語であるPrologではホーン節で述語を定義することを基本にしており、すべての1階述語を扱えるわけではない。ホーン節に含まれない式を扱うために拡張された機能もあるが、それは論理の枠を超えたものである。一方では、単一化可能を表す述語=を基本述語とする公理系に関する研究が進み、限量子除去法によって、与えられた任意の式を選言標準形に変形するアルゴリズムが発表されている。また、その公理系にユーザ定義述語を加えた系における、述語の展開に関する研究でも次項で説明するような結果が得られている。

本稿ではこれらの結果をもとに、完備化されたユーザ定義述語のもとで、自由変数を含む任意の式に対して「逐次漸近標準形」と呼ぶ式が計算できることを示し、それを実際に実現した例を紹介する。この変形は、計算量の問題はあるもののホーン節に限定されない任意の1階述語言語のインタプリタと考えることもできる。

2. 準備

本稿では特に指示のない限り、x、y、z、u、vは変数を、 f、gは関数名を、a、b、eは式を、p、qは論理式を表すことにする。これらに添字が付いたものも同じ型と考える。本稿では=は後で述べるように単一化可能を意味する述語を示すために使用し、通常の意味の等しいことを表すためには≡を使用する。文字をボールド体にしたものはその文字の型のデータの「並び」を表すことにする。すなわち、**a**は$a_1,a_2,...,a_n$のことであり、**a**=**b**は要素の数が等しくないときは偽であり、等しいときは $a_1=b_1 \wedge ... \wedge a_m=b_m$ であることを表すものとする。並びは文脈によっては集合として考えることもあり、そのときには通常のローマン体の文字は1要素からなる集合と考える。並びの連結は「a,**b**」のようにカンマで区切ることで示す。

式に含まれている自由変数を明示的に示すために、$a[\mathbf{x}]$、$a[\mathbf{x},...]$、$a[\hat{\mathbf{x}}]$という記法を使用する。この意味は、式a中に含まれる明示的な自由変数の集合を$\mathbf{V}_f(a)$としたときに、それぞれ$\mathbf{x}\equiv\mathbf{V}_f(a)$、$\mathbf{x}\subset\mathbf{V}_f(a)$、$\mathbf{x}\cap\mathbf{V}_f(a)\equiv\emptyset$である。

Σを関数記号の集合としたときに単一化可能を表わす述語$=$に関する公理系$\mathrm{E}(\Sigma)$を考える[1]:

$$\mathrm{E}(\Sigma)\equiv\{f(\mathbf{a})=f(\mathbf{b})\rightarrow\mathbf{a}=\mathbf{b}\ ,\ \neg f(\mathbf{a})=g(\mathbf{b})\ ,\ \neg x=e[x,...]\ \mid\ f,g\in\Sigma\ ,\ f\neq g\}$$

Σ中の関数記号と$=$から作られる自由変数を含む一階述語論理式の全体を$\mathrm{F}(\Sigma,=)$とすると、$|\Sigma|=\infty$のときには$\mathrm{F}(\Sigma,=)$中の任意の式はそれと同値な選言標準形 DNF と呼ばれる式に変換できることが知られている[2][3][6]。 ($|\Sigma|<\infty$のときにもΣにある制約を加えれば上のことが言える。) 元の式中の自由変数を$\mathbf{x}$としたときにDNFは以下に示す形をしている:

$$\mathrm{DNF}\equiv\Psi_1\vee\Psi_2\vee...\vee\Psi_m\ ,\ (m\geq0)$$
$$\Psi_i\equiv\alpha_{i0}\wedge\neg\alpha_{i1}\wedge...\wedge\neg\alpha_{in_i}\ ,\ (n_i\geq0)$$
$$\alpha_{ij}\equiv\exists\mathbf{y}_{ij}.\mathbf{x}_{ij}=\mathbf{e}_{ij}[\mathbf{y}_{ij}]\ ,\ (\mathbf{x}_{ij}\cap\mathbf{y}_{ij}\equiv\emptyset)$$

以下ではΨ_iのことを連言式、α_{ij}のことを ($\mathbf{x}_{ij}$に関する)基礎式と呼ぶことにする。また$\mathrm{F}(\Sigma,=)$中の式pの選言標準形のことを$\mathrm{DNF}(p)$と書くことにする。なお、DNF 中の各連言式は Prolog における1組の解に相当しており (Prolog の場合には連言式は1つの基礎式だけからなっている($n_i\equiv0$)。)、 $\mathrm{DNF}(p)$はpのすべての解の集合と考えることができる。

一方、ユーザ定義述語が加えられた場合については次のことが知られている。完備化した述語定義をP^*と書き、$\models_3$を3値論理における論理的帰結関係[4]とすると、$=$、Σ、P^*から構成される、自由変数を含まない任意の論理式Gについて、ある整数nが存在して次の2つの式が同値になる[4]。

(2.1) $\mathrm{E}(\Sigma),\mathrm{P}^*\models_3\mathrm{G}$

(2.2) $\mathrm{E}(\Sigma)\models_3\mathrm{G}^{(n)}\{\mathrm{U};\mathrm{U}\}$

なお上の式で$\mathrm{G}^{(n)}$は以下のように定義される。

$\mathrm{G}^{(0)}\equiv\mathrm{G}$,
$\mathrm{G}^{(n+1)}\equiv\mathrm{G}^{(n)}$中のすべての$=$以外の述語を1回だけ*unfold*した式

また$\mathrm{G}^{(n)}\{e_+;e_-\}$は$\mathrm{G}^{(n)}$中の $=$以外の述語のうちトップレベルから見たときに肯定で現れ

るものにはe_+を、否定で現れるものにはe_-を代入した式であり、Uは3値のうち未定値を表わす記号である。さらに$\models_2$を通常の2値論理における論理的帰結関係としたとき，(2.2)式と

(2.3) $E(\Sigma)\models_2 G^{(n)}\{F;T\}$

とが同値であることが示されており[5], (2.1)式は結局2値論理での問題に還元される.

(2.1)式は3値論理についての言明であるが、2値論理との関係は次のようになる。すなわち、述語の定義とGが以下で説明するような厳密な場合には3値で考えた場合と2値で考えた場合が同値である[4]. 言い換えれば、 P*が厳密であり、Gがgroundリテラルであるときには(2.1)と

(2.4) $E(\Sigma),P^*\models_2 G$

は同値である。ここで述語定義が厳密であるとは, ある述語から他の述語を呼ぶときに常に肯定あるいは常に否定で呼んでいるような定義である. なお, この呼び出しには間接的なものも含まれている.

結局、P*が厳密でGがgroundリテラルのときには

(2.5) $E(\Sigma),P^*\models_2 G \Leftrightarrow \exists n.E(\Sigma)\models_2 G^{(n)}\{F;T\}$

であり、それ以外の場合にも

(2.5′) $E(\Sigma),P^*\models_3 G \Leftrightarrow \exists n.E(\Sigma)\models_2 G^{(n)}\{F;T\}$

は言える。

3. 逐次漸近標準形

ユーザ定義述語P*のもとで、自由変数xを含む任意の式Gを考える。xの値によってGの値が定まるので G(x)と書くことにする。 (2.5')より、任意の定数aに対して次の2つの式は同値である。

(3.1) $E(\Sigma),P^*\models_3 G(a)$

(3.2) $\exists n.E(\Sigma)\models_2 G(a)^{(n)}\{F;T\}$

ここで

$G_n(\mathbf{x}) \equiv G(\mathbf{x})^{(n)}\{F;T\}$

と定義すると、任意のaについて(3.1)と

$\exists n.(E(\Sigma) \models_2 G_n(\mathbf{a}))$

は同値である。このとき$G_n(\mathbf{x})$という式は Σと=だけからなる式であるので前節で述べたように選言標準形にすることができる。そこで

$Anorm(G(\mathbf{x}),n) \equiv DNF(G_n(\mathbf{x}))$

と定義し、これを(n次の)漸近標準形と呼ぶことにする。漸近と呼ぶのは$Anorm(G(\mathbf{x}),n)$が以下の性質を持つからである。

$E(\Sigma) \models_2 Anorm(G(\mathbf{x}),n) \rightarrow Anorm(G(\mathbf{x}),n+1)$

$E(\Sigma),P^* \models_2 Anorm(G(\mathbf{x}),n) \rightarrow G(\mathbf{x})$

$\forall \mathbf{a}.(E(\Sigma),P^* \models_3 G(\mathbf{a})\ ) \rightarrow (\exists n.E(\Sigma) \models_2 Anorm(G(\mathbf{a}),n)\ )$

すなわち、各nについて$Anorm(G(\mathbf{x}),n)$を満たす$\mathbf{x}$は $Anorm(G(\mathbf{x}),n+1)$および$G(\mathbf{x})$を満たし、3値論理で$G(\mathbf{x})$を満たす$\mathbf{x}$は、十分大きなnをとれば$Anorm(G(\mathbf{x}),n)$を満たしている。(なお、最後の式は述語定義とGが「厳密」な場合には2値でも成り立つ。)

上記の漸近標準形はnを与えて始めて計算できる式であり、 nを増加させながら逐次的に「答え」を求めるわけではない。逐次的に求めるには次のようにする。すなわち、

$R_0(\mathbf{x}) \equiv G(\mathbf{x})$
$R_{n+1}(\mathbf{x}) \equiv \neg H_n(\mathbf{x}) \wedge R_n^{(1)}(\mathbf{x})$
$H_n(\mathbf{x}) \equiv R_n(\mathbf{x})\{T;F\}$

と定義すると、次の関係が成り立つ:

$G^{(n)}(\mathbf{x}) \leftrightarrow H_0(\mathbf{x}) \vee \ldots \vee H_{n-1}(\mathbf{x}) \vee R_n(\mathbf{x})$

これより、

$G_n(\mathbf{x}) \leftrightarrow H_0(\mathbf{x}) \vee \ldots \vee H_n(\mathbf{x})$

となるので、$DNF(H_i(\mathbf{x}))$を次々に求めれば「解」が逐次的に求まる。この
$H_0(\mathbf{x}) \vee \ldots \vee H_n(\mathbf{x})$ を逐次漸近標準形と呼ぶ。

4. 逐次漸近標準変形の実現

漸近標準形への変形はまず、$H_n(\mathbf{x})$ を求めて次にこれを選言標準形DNFに変形することにする。最初の変形は普通の展開であり、特に述べることもないので、ここではDNFへの変形について述べる。

方針は、式の内側から順番にDNFへの変形を行なって、最後に全体にDNF化が終了するようにする。そこで、=の式をDNFに変形する演算と各論理結合子に対応するDNFどうしの演算を用意すればよいことになる。

=の式から求まるDNFはそれが偽でないときには、1つの連言式からなり、しかもその連言式は否定を含まない1つの基礎式から成っている。 =の両辺のどちらかが変数の場合には、$\neg x=e[x,...]$によって偽になるか、あるいは基礎式になる。偽にならない場合の式を $x=e[\mathbf{z}]$とする。このときの$x,\mathbf{z}$に関する基礎式は

$$\exists \mathbf{y}.(x,\mathbf{z}=e[\mathbf{z}]\{\mathbf{y}/\mathbf{z}\},\mathbf{y})$$

となる。この式で、$a,\mathbf{b}$という表記は、並び$\mathbf{b}$にaを連結した並びを示している。また$\mathbf{y}$は$\mathbf{z}$に対応した新たな変数である。両辺のどちらも変数ではないときには、$\neg f(\mathbf{a})=g(\mathbf{b})$ によって偽になるか、$f(\mathbf{a})=f(\mathbf{b}) \rightarrow \mathbf{a}=\mathbf{b}$ によって分解され、その式に変形手続きを適用することになる。

各論理結合子に対応するDNFの演算では、連言式、基礎式についての演算を使用するのでまずそれを述べる。

基礎式についての演算では、まず式の自由変数の集合を同じものにする。すなわち、2つの基礎式を $\exists \mathbf{y}_1.\mathbf{x}_1=\mathbf{e}_1$ と $\exists \mathbf{y}_2.\mathbf{x}_2=\mathbf{e}_2$ であるとすると、 $\mathbf{x}\equiv\mathbf{x}_1\bigcup\mathbf{x}_2$として、それぞれを$\exists \mathbf{y}_1'.\mathbf{x}=\mathbf{e}_1'$、 $\exists \mathbf{y}_2'.\mathbf{x}=\mathbf{e}_2'$と変形する。以下ではこの処理が暗黙に行なわれているものとする。基礎式の否定は単純で、$\exists \mathbf{y}.\mathbf{x}=\mathbf{e}$は $\neg\exists \mathbf{y}.\mathbf{x}=\mathbf{e}$に、 $\neg\exists \mathbf{y}.\mathbf{x}=\mathbf{e}$は $\exists \mathbf{y}.\mathbf{x}=\mathbf{e}$にするだけである。 $\exists \mathbf{y}_1.\mathbf{x}=\mathbf{e}_1 \wedge \exists \mathbf{y}_2.\mathbf{x}=\mathbf{e}_2$の演算は、 $\mathbf{e}_1$と$\mathbf{e}_2$の最小汎化式(most general expression)$\mathbf{e}$を求めて、$\exists \mathbf{y}_1\bigcup\mathbf{y}_2.\mathbf{x}=\mathbf{e}$という式から冗長な変数を消去して簡単化したものを結果とする。

$\exists \mathbf{y}_1.\mathbf{x}=\mathbf{e}_1 \wedge \neg\exists \mathbf{y}_2.\mathbf{x}=\mathbf{e}_2$のときは、 $\mathbf{e}_1$を右辺の式の$\mathbf{x}$に代入して簡単化を行ない、偽になれば結果は偽であり、そうでないときには2つの式の論理積にする。

$\neg\exists \mathbf{y}_1.\mathbf{x}=\mathbf{e}_1 \wedge \neg\exists \mathbf{y}_2.\mathbf{x}=\mathbf{e}_2$のときは、次に示すような基礎式間の半順序関係$<_g$を調べる述語を使用する。ここで$<_g$とは、左辺の式が右辺の式の特殊化になっているという関係、すなわち、右辺への代入で左辺に等しくなるものがあるかどうかという関係である。これは単一化よりは簡単に調べることができる。一般に、$\mathbf{e}_1<_g\mathbf{e}_2$のときには、 $\neg\exists \mathbf{y}_1.\mathbf{x}=\mathbf{e}_1 \wedge \neg\exists \mathbf{y}_2.\mathbf{x}=e_2 \Leftrightarrow \neg\exists \mathbf{y}_2.\mathbf{x}=e_2$ という関係があるのでこれを利用すると、元の式の両辺のどちらかが不要になることがある。基礎式についての演算のうち最初のもの

以外は冗長なものの消去であり、必須のものではないが式を見易くするための簡単化である。

連言式については基礎式についての演算を利用して同様に処理するが、簡単化は、偽になる式の認識と冗長な式の消去だけを行なっており、「最も簡単な」式を求めているわけではない。連言式の場合にも、基礎式の場合と同様に $\Psi_1 <_g \Psi_2$ という半順序関係を、基礎式に対する半順序関係から定義して、それを利用している。

DNFについては、基礎式、連言式についての上の演算を利用して各論理結合子についての演算を実現している。

限量子の消去については、[6]の方法を使用した。 [6]では任意の連言式Ψについて$\exists y.\Psi$をDNFにする方法が示されているが、以下の関係を用いればこれで十分である。

$$\exists \mathbf{y}.(p \vee q) \Leftrightarrow (\exists \mathbf{y}.p) \vee (\exists \mathbf{y}.q)$$
$$\forall \mathbf{y}.p \Leftrightarrow \neg \exists \mathbf{y}.\neg p$$

まず連言式中の各基礎式における自由変数を上記と同様にして揃えた上で次のように変形する:

$$\begin{aligned}\Psi &\equiv \exists \mathbf{z}_0.\mathbf{x}=\mathbf{e}_0[\mathbf{z}_0] \wedge \neg\exists \mathbf{z}_1.\mathbf{x}=\mathbf{e}_1[\mathbf{z}_1] \wedge \ldots \wedge \neg\exists \mathbf{z}_n.\mathbf{x}=\mathbf{e}_n[\mathbf{z}_n] \\ &\leftrightarrow \exists \mathbf{z}_0.(\mathbf{x}=\mathbf{e}_0[\mathbf{z}_0] \wedge \neg\exists \mathbf{z}_1.\mathbf{e}_0[\mathbf{z}_0]=\mathbf{e}_1[\mathbf{z}_1] \wedge \ldots \wedge \neg\exists \mathbf{z}_n.\mathbf{e}_0[\mathbf{z}_0]=\mathbf{e}_n[\mathbf{z}_n]) \\ &\leftrightarrow \exists \mathbf{z}_0.(\mathbf{x}=\mathbf{e}_0[\mathbf{z}_0] \wedge \neg\exists \mathbf{z}_1.\mathbf{z}_0=\mathbf{e}'_1[\mathbf{z}_1] \wedge \ldots \wedge \neg\exists \mathbf{z}_n.\mathbf{z}_0=\mathbf{e}'_n[\mathbf{z}_n])\end{aligned}$$

ここで$\mathbf{x}$を$y,\mathbf{x}'$とし、それに対応して $\mathbf{e}_0[\mathbf{z}_0]$を $a[\mathbf{u}_0,\mathbf{v}_0],\mathbf{e}'_0[\mathbf{u}_0]$ と書くと($\mathbf{v}_0 \equiv \mathbf{z}_0-\mathbf{u}_0$)、

$$\begin{aligned}\exists y.\Psi \leftrightarrow \exists y.(\exists \mathbf{u}_0,\mathbf{v}_0.(y,\mathbf{x}'=a[\mathbf{u}_0,\mathbf{v}_0],\mathbf{e}'_0[\mathbf{u}_0] \wedge \neg\exists \mathbf{u}_1,\mathbf{v}_1.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_1[\mathbf{u}_1],\mathbf{c}_1[\mathbf{u}_1,\mathbf{v}_1] \wedge \\ \ldots \wedge \neg\exists \mathbf{u}_n,\mathbf{v}_n.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_n[\mathbf{u}_n],\mathbf{c}_n[\mathbf{u}_n,\mathbf{v}_n])) \\ \leftrightarrow \exists \mathbf{u}_0.(\mathbf{x}'=\mathbf{e}'_0[\mathbf{u}_0] \wedge \exists \mathbf{v}_0.(\neg\exists \mathbf{u}_1,\mathbf{v}_1.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_1[\mathbf{u}_1],\mathbf{c}_1[\mathbf{u}_1,\mathbf{v}_1] \wedge \\ \ldots \wedge \neg\exists \mathbf{u}_n,\mathbf{v}_n.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_n[\mathbf{u}_n],\mathbf{c}_n[\mathbf{u}_n,\mathbf{v}_n]))\end{aligned}$$

この式で$\mathbf{c}_k[\mathbf{u}_k,\mathbf{v}_k] \equiv \mathbf{v}_k$の項については

$$\exists \mathbf{u}_k,\mathbf{v}_k.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_k[\mathbf{u}_k],\mathbf{v}_k \leftrightarrow \exists \mathbf{u}_k.\mathbf{u}_0=\mathbf{b}_k[\mathbf{u}_k] \wedge \exists \mathbf{v}_k.\mathbf{v}_0=\mathbf{v}_k \leftrightarrow \exists \mathbf{u}_k.\mathbf{u}_0=\mathbf{b}_k[\mathbf{u}_k]$$

となるので

$$\begin{aligned}\exists y.\Psi \leftrightarrow \exists \mathbf{u}_0.(\mathbf{x}'=\mathbf{e}'_0[\mathbf{u}_0] \wedge \neg\exists \mathbf{u}_1.\mathbf{u}_0=\mathbf{b}_1[\mathbf{u}_1] \wedge \ldots \wedge \neg\exists \mathbf{u}_m.\mathbf{u}_0=\mathbf{b}_m[\mathbf{u}_m] \wedge \\ \exists \mathbf{v}_0.(\neg\exists \mathbf{u}_{m+1},\mathbf{v}_{m+1}.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_{m+1}[\mathbf{u}_{m+1}],\mathbf{c}_{m+1}[\mathbf{u}_{m+1},\mathbf{v}_{m+1}] \wedge \\ \ldots \wedge \neg\exists \mathbf{u}_n,\mathbf{v}_n.\mathbf{u}_0,\mathbf{v}_0=\mathbf{b}_n[\mathbf{u}_n],\mathbf{c}_n[\mathbf{u}_n,\mathbf{v}_n]))\end{aligned}$$

また、$\mathbf{c}_k[\mathbf{u}_k,\mathbf{v}_k]$が$\mathbf{v}_k$ とは異なる部分では$\exists \mathbf{v}_0 \ldots$を満たす$\mathbf{v}_0$ が存在するので[6]

$$\exists y.\Psi \leftrightarrow \exists \mathbf{u}_0.(\mathbf{x}' = \mathbf{e}_0'[\mathbf{u}_0] \wedge \neg\exists \mathbf{u}_1.\mathbf{u}_0 = \mathbf{b}_1[\mathbf{u}_1] \wedge \ldots \wedge \neg\exists \mathbf{u}_n.\mathbf{u}_0 = \mathbf{b}_m[\mathbf{u}_m])$$
$$\leftrightarrow \exists \mathbf{u}_0.(\mathbf{x}' = \mathbf{e}_0'[\mathbf{u}_0] \wedge \neg\exists \mathbf{u}_1.\mathbf{e}_0'[\mathbf{u}_0] = \mathbf{e}_0'[\mathbf{u}_0]\{\mathbf{u}_0{:}\mathbf{b}_1[\mathbf{u}_1]\}$$
$$\wedge \ldots \wedge \neg\exists \mathbf{u}_m.\mathbf{e}_0'[\mathbf{u}_0] = \mathbf{e}_0'[\mathbf{u}_0]\{\mathbf{u}_0{:}\mathbf{b}_1[\mathbf{u}_m]\})$$
$$\leftrightarrow \exists \mathbf{u}_0.\mathbf{x}' = \mathbf{e}_0'[\mathbf{u}_0] \wedge \neg\exists \mathbf{u}_1.\mathbf{x}' = \mathbf{e}_0'[\mathbf{u}_0]\{\mathbf{u}_0{:}\mathbf{b}_1[\mathbf{u}_1]\} \wedge \ldots \wedge \neg\exists \mathbf{u}_m.\mathbf{x}' = \mathbf{e}_0'[\mathbf{u}_0]\{\mathbf{u}_0{:}\mathbf{b}_1[\mathbf{u}_m]\})$$

となって標準形になる。

5. 例

Lispで作成した上記の漸近標準変形を以下に示す。述語定義は(define (<name> <args>) <body>)で、それ以外の式には逐次漸近標準形を「;」が入力されるたびに次数を上げて求める。まず、evenを定義し、これから逐次標準形を3次まで求めたものを示す。(なおこのevenの定義は「厳密」ではない。)

```
=> (define (even x)
     (or (= x (0))
         (some (y) (and (= x (s y)) (not (even y))))))
even
=> (even a)
H0:(false);
H1:(= a (0));
H2:(and (some (?0) (= a (s ?0)))
        (not (= a (s (0))))
        (not (some (?0) (= a (s (s ?0))))));
H3:(= a (s (s (0))))
```

ここでは、0を示す(0)や2を示す(s (s (0)))が解に含まれている。

次の例は素数を求めるものである。まず、加算と除算を、次に素数を定義しているが、ここでの素数の定義は、「2以上で、かつ、約数があればそれは1か自分自身である自然数」という数学本来のものである。

```
=> (define (add x y z)
     (or (and (= x (0)) (= y z))
         (some (p q) (and (= x (s p)) (= z (s q)) (add p y q)))))
=> (define (div d a)
     (and (some (x) (= d (s x)))
```

```
          (or (= a (0))
            (some (u) (and (add d u a) (div d u))))))
=> (define (prime a)
     (and (some (x) (= a (s (s x))))
          (any (d) (imply (div d a) (or (= d (s (0))) (= d a))))))
=> (prime x)
H0:(false);
H1:(false);
H2:(false);
H3:(false);
H4:(false);
H5:(and (some (?0) (= x (s (s ?0))))
        (not (some (?0) (= x (s (s (s ?0))))))))
```

この実行では、primeの引数が「自然数」以外のものを否定していないので、自然数に限定してみる。

```
=> (define (number a)
     (or (= a (0))
         (some (x) (and (= a (s x)) (number x)))))
=> (and (prime x) (number x))
H0:(false);
H1:(false);
H2:(false);
H3:(false);
H4:(false);
H5:(= x (s (s (0))));
H6:(= x (s (s (s (0)))));
H7:(false);
H8:(= x (s (s (s (s (s (0)))))));
H9:(false);
H10:(= x (s (s (s (s (s (s (s (0)))))))))
```

10次までの逐次漸近標準形を求めると2、3、5、7が素数として求まっている。以上の素数の定義は「厳密」なものであるので、次数を上げていけば素数を数え上げることができる。

6. 考察

E(Σ)にユーザ定義述語を加えたシステムにおいて、自由変数を含む任意の論理式から逐次漸近標準形と呼ぶことのできる形に変形すること、およびその実現例を示した。逐次漸近標準形の各連言式はPrologにおける解を拡張した形をしており、逐次漸近標準変形は、ホーン節に限らない一般の一階述語言語のインタプリタと考えることができる。

ただし、このままではある次数ですべての解が求まったとしても、そのことを知る手段がない。これに関しては$R_n(\mathbf{x})$を部分評価してFになれば、それ以上の解がないことがわかるので、現在実現を試みている。

本稿では任意の一階述語論理の式から解を求めることが可能であることを示したが、逐次漸近標準変形を汎用の一階述語言語と考えるには計算量の上で問題があり、これを直接プログラミング言語として使用するにはこのままでは無理があると思われる。

文献

[1] Clark,K.L., "Negation as Failure", in Logic and Database, Plenum Press, New York, 1978.

[2] Comon,H. and Lescanne,P., "Equational Problems and Disunification", J. Symbolic Computation, Vol 7, pp371-425, 1989.

[3] Kunen,K., "Answer Sets and Negation as Failure", Proc. 4th Int. Conf. on Logic Programming, Melbourne, pp219-228, 1987.

[4] Kunen,K., "Signed Data Dependencies in Logic Programs", Computer Science Technocal Report 719, Univ. of Wisconsin-Madison, 1987.

[5] Sato,T., "A First Order Unfold/Fold System", ETL Technical Reprt TR-90-17, Tsukuba, 1990.

[6] Sato,T., "Quantifier Elimination for Finite and Infinite Trees", ETL Technical Reprt TR-89-25, Tsukuba, 1989.